The interface models the object as a set of input streams and a set of output streams each identified by a 0-based index.

The DMO_MEDIA_TYPE structure is used to identify all types of multimedia data.

IMediaObject methods and structures

Data buffers are wrapped inside an IMediaBuffer interface, a pointer to which is stored in each of the below structures. The IMediaBuffer interface contains a few simple methods to access the data pointer and the buffer's length:

```
interface IMediaBuffer : IUnknown
{
   HRESULT SetLength(
     [in] DWORD cbLength
   );
   HRESULT GetMaxLength(
     [out] DWORD *pcbMaxLength
   );
   HRESULT GetBufferAndLength(
     [out] BYTE **ppBuffer,  // not filled if NULL
     [out] DWORD *pcbLength // not filled if NULL
   );
}
```

DMO_MEDIA_TYPE

```
        {
                GUID    majortype;
                GUID    subtype;
                BOOL     bFixedSizeSamples;
                BOOL     bTemporalCompression;
                ULONG    lSampleSize;
                GUID    formattype;
                IUnknown *pUnk;
                ULONG    cbFormat;
                BYTE *pbFormat;
        }
```

This defines the media type structure. This structure exactly matches the DirectShow AM_MEDIA_TYPE structure and is given here for reference.

DMO_OUTPUT_DATA_BUFFER

```
{
        IMediaByffer          *pBuffer;
        DWORD                 dwFlags;
        REFERENCE_TIME        rtTimeStamp;
        REFERENCE_TIME        rtTimeLength;
}
```

pBuffer	Pointer to buffer wrapper interface Can be NULL
dwFlags (out)	

This must be a combination of the following flag values (or 0)

DMO_OUTPUT_DATA_BUFFERF_TIME
rtTimestamp is valid

DMO_OUTPUT_DATA_BUFFERF_TIMELENGTH
rtTimeLength is valid

DMO_OUTPUT_DATA_BUFFERF_INCOMPLETE
Another buffer is required to continue process the input

DMO_OUTPUT_DATA_BUFFERF_SYNCPOINT
Syncpoint at the beginning of the data[1]

rtTimestamp	Start time
rtTimelength	Length

HRESULT GetStreamCount(DWORD *pcInputStreams, DWORD *pcOutputStreams)

Returns the number of each type of stream. It is possible for objects to have no input streams or no output streams.

Parameters

pcInputStreams	Number of input streams
pcOutputStreams	Number of output streams

Return Values

S_OK	Success
E_POINTER	One of the parameters was NULL
Failure code	Some other failure

[1] Implicit syncpoints (e.g., regularly occurring syncpoints in an audio stream) must be reflected by this flag. In addition, audio buffers must always begin at a syncpoint.

HRESULT GetInputStreamInfo(DWORD dwInputStreamIndex, DWORD *pdwFlags)

Returns information about an input stream. This information does not change for the lifetime of this stream.

Parameters

dwInputStreamIndex	Zero based index of input stream
pdwFlags	DMO_INPUT_INFOF_HOLDS_BUFFERS The Media Object may hold on to multiple input buffers for this stream

Return Values

S_OK	Success
E_POINTER	**pdwFlags** was NULL
DMO_E_INVALID_STREAM	Stream index out of range
Failure code	Some other failure

HRESULT GetOutputStreamInfo(DWORD dwOutputStreamIndex, DWORD *pdwFlags)

Returns information about an output stream. This information does not change for the lifetime of this stream.

Parameters

dwOutputStreamIndex	Zero based index of output stream
pdwFlags	DMO_OUTPUT_STREAMF_WHOLE_SAMPLES Output contains complete samples
	DMO_OUTPUT_STREAMF_SINGLE_SAMPLE Output contains 1 sample
	DMO_OUTPUT_STREAMF_FIXED_SAMPLE_SIZE Output samples are fixed size
	DMO_OUTPUT_STREAMF_DISCARDABLE Output can be discarded in **ProcessOutput**
	DMO_OUTPUT_STREAMF_OPTIONAL Processing this stream is optional

Return Values

S_OK	Success
E_POINTER	**pdwFlags** was NULL
DMO_E_INVALID_STREAM	Stream index out of range
Failure code	Some other failure

HRESULT GetInputType(DWORD dwInputStreamIndex, DWORD dwTypeIndex, DMO_MEDIA_TYPE *pmt)

Get the **dwTypeIndex** type for the input stream **dwInputStreamIndex**. The Media Type returned in pmt will be overwritten if the method is successful. The format block of the Media Type must be freed by calling **CoTaskMemFree()**. Often input types will contain a NULL format block.

The types are enumerated in preference order with the most preferred type corresponding to a type index of 0.

If the call is successful the caller should free the output media type by calling **MoFreeMediaType()** when it's done.

For convenience of implementation it's possible that some types will be enumerated which will fail when used in **SetOutputType()**.

Parameters

dwInputStreamIndex	Zero based input stream index
dwTypeIndex	Zero based type index
pmt	Pointer to Media type to return

Return Values

S_OK	Success
S_FALSE	Type index out of range
DMO_E_INVALID_STREAM	Stream index out of range
E_OUTOFMEMORY	Could not allocate format block or some other memory failure
Failure code	Some other failure

HRESULT GetOutputType(DWORD dwOutputStreamIndex, DWORD dwTypeIndex, DMO_MEDIA_TYPE *pmt)

Get the **dwTypeIndex** type for the output stream **dwOutputStreamIndex**. The Media Type returned in pmt will be overwritten if the method is successful. The format block of the Media Type must be freed by calling **CoTaskMemFree()**.

The types are enumerated in preference order with the most preferred type corresponding to a type index of 0.

If the input type is not set for some input stream this call may fail or return a type with a NULL format block.

If the call is successful the caller should free the output media type by calling **MoFreeMediaType()** when it's done.

For convenience of implementation it's possible that some types will be enumerated which will fail when used in **SetOutputType()**.

Parameters

dwOutputStreamIndex	Zero based output stream index
dwTypeIndex	Zero based type index
pmt	Pointer to Media type to return

Return Values

S_OK	Success
S_FALSE	Type index out of range
DMO_E_INVALID_STREAM	Stream index out of range

E_OUTOFMEMORY	Could not allocate format block or some other memory failure
Failure code	Some other failure

HRESULT SetInputType(DWORD dwInputStreamIndex, const DMO_MEDIA_TYPE *pmt, DWORD dwFlags)

Set a Media Type for an input stream. This call is processed in the context of the types currently set for other streams.

Parameters

dwInputStreamIndex	Zero based input stream index
pmt	Type to set
dwFlags	This must be a combination of the following flag values (or 0) DMO_SET_TYPEF_TEST_ONLY Just check if this type can be set, do not set it DMO_SET_TYPEF_CLEAR Clears the type so that no type is set for this stream

Return Values

S_OK	Type was set
S_FALSE	Type cannot be set
DMO_E_TYPE_NOT_ACCEPTED	Type is not acceptable
Failure code	Some other failure
E_INVALIDARG	Invalid argument

HRESULT SetOutputType(DWORD dwOutputStreamIndex, const DMO_MEDIA_TYPE *pmt, DWORD dwFlags)

Set a Media Type for an output stream. This call is processed in the context of the types currently set for other streams.

Parameters

dwOutputStreamIndex	Zero based output stream index
pmt	Type to set
dwFlags	This must be a combination of the following flag values (or 0) DMO_SET_TYPEF_TEST_ONLY Just check if this type can be set, do not set it DMO_SET_TYPEF_CLEAR Clears the type so that no type is set for this stream

Return Values

S_OK	Type can be set

S_FALSE	Type cannot be set
DMO_E_TYPE_NOT_ACCEPTED	Type is not acceptable
Failure code	Some other failure

HRESULT GetInputCurrentType(DWORD dwInputStreamIndex, DMO_MEDIA_TYPE *pmt)

Get the Media Type for an input stream.

If the call is successful the caller should free the output media type by calling **MoFreeMediaType()** when it's done.

Parameters

dwInputStreamIndex	Zero based input stream index
pmt	Type returned here The type must be freed by calling **MoFreeMediaType()** if this call was successful

Return Values

S_OK	Type was returned
DMO_E_TYPE_NOT_SET	Type is not set
E_OUTOFMEMORY	Format block could not be allocated
Failure code	Some other failure

HRESULT GetOutputCurrentType(DWORD dwOutputStreamIndex, DMO_MEDIA_TYPE *pmt)

Get the Media Type for an output stream.

If the call is successful the caller should free the output media type by calling **MoFreeMediaType()** when it's done.

Parameters

dwOutputStreamIndex	Zero based output stream index
pmt	Type returned here The type must be freed by calling **MoFreeMediaType()** if this call was successful

Return Values

S_OK	Type was be set
DMO_E_TYPE_NOT_SET	Type is not set
E_OUTOFMEMORY	Format block could not be allocated
Failure code	Some other failure

HRESULT GetInputSizeInfo(DWORD dwInputStreamIndex, DWORD *pcbSize, DWORD *pcbMaxLookahead, DWORD *pdwAlignment)

Get buffer size and alignment requirements for a given input stream.

This method should be called after the types of all streams have been set using **SetInputType()** and **SetOutputType()**.

pcbMaxLookahead is only used for objects which hold on to multiple input buffers for lookahead. In that case the application must allow for enough buffers so that this amount can be retained by the object in order to generate output. For example, if the application decides on a fixed buffer size of **dwBufferSize** then it should be prepared to allocate up to at least:

(*pcbMaxLookahead + 2 * (dwBufferSize - 1)) / dwBufferSize

buffers of that size to avoid running out of buffers. This number may be reduced if there are alignment requirements in the data.

Parameters

dwInputStreamIndex	Zero based input stream index
pcbSize	Returns buffer size This is at least the minimum size required for processing
pcbMaxLookahead	Maximum size of data held by this object if it AddRefs multiple input buffers Or 0.
pdwAlignment	Returns buffer alignment. 1 means no alignment requirement.

Return Values

S_OK	Call successful
E_POINTER	NULL pointer passed in
Failure code	Other failure

09625269.071300

HRESULT GetOutputSizeInfo(DWORD dwOutputStreamIndex, DWORD *pcbSize, DWORD *pdwAlignment)

Get buffer size and alignment requirements for a given output stream.

This method should be called after the types of all streams have been set using **SetInputType()** and **SetOutputType()**.

Parameters

dwOutputStreamIndex	Zero based output stream index
pcbSize	Returns buffer size
pdwAlignment	Returns buffer alignment. 1 means no alignment requirement.

Return Values

S_OK	Call successful
E_POINTER	NULL pointer passed in

Failure code	Other failure

HRESULT Discontinuity(DWORD dwInputStreamIndex)

Informs the Media Object that the data is discontinuous on input stream **dwInputStreamIndex**. This can occur (for example) because there is a large gap in the data, because no more data is expected, or because the format of the data is changing.

The Media Object should generate all output which can be generated from the data already received in calls to **ProcessInput()** on this stream before accepting more data on this stream.

Calling **Discontinuity()** more than once without intermediates call to **ProcessInput()** is equivalent to calling **Discontinuity()** once.

If the **Discontinuity()** method has been called on all input streams for a Media Object and all output has been processed from all the output streams by calls to **ProcessOutput()** then the Media Object is in the equivalent state to the flushed state. In this state all buffers must be released and no more output can be generated until **ProcessInput()** is called again. Also in this state calling **Flush()** has no effect.

Parameters

dwInputStreamIndex	0-based input stream index

Return Values

S_OK	Call was successful

HRESULT Flush()

Flush internally buffered data and reset any streaming state. Media types and other parameters such as latency are not changed.

All Media Buffers held by all streams must be released on return from this call. Any incomplete processing of a discontiuity for any input stream is cancelled.

All streams should accept input after a **Flush()** call.

Parameters

Return Values

S_OK	Successful
Failure code	Failure

HRESULT AllocateStreamingResources()

Hint to allocate any resources necessary for processing. This method may not be called before the first call to **ProcessOutput()** and it is not required to support this method.

Parameters

Return Values

S_OK	Successful. Return this if the call is not implemented.
Failure code	Some failure occurred

HRESULT FreeStreamingResources()

Hint to free any resources necessary for processing.

Parameters

Return Values

S_OK	Successful. Return this if the call is not implemented.
Failure code	Some failure occurred

HRESULT GetInputStatus(DWORD dwInputStreamIndex, DWORD *pdwFlags)

Return input stream status.

Parameters

dwInputStreamIndex	0-based input stream index
pdwFlags	DMO_INPUT_STATUSF_ACCEPT_DATA This stream is ready to accept data

The setting of this flag can only change as the result of one of the following calls:

ProcessOutput()
Discontinuity()
ProcessInput()
Flush()

Return Values

HRESULT ProcessInput(DWORD dwInputStreamIndex, IMediaBuffer *pBuffer, DWORD dwFlags, REFERENCE_TIME rtTimeStamp, REFERENCE_TIME rtTimeLength)

Deliver an input buffer for a stream. The Media Object should either process all the data inside this method or call **IMediaBuffer::AddRef()** to hold the buffer waiting for calls to **ProcessOutput()**. When the Media Object has generated all the output it can from this buffer it should call **IMediaBuffer::Release()** unless it needs the buffer for lookahead.

If the Media Object calls **IMediaBuffer::AddRef()** the appication should not reuse a buffer until the matching **IMediaBuffer::Release()** is called.

Parameters

dwInputStreamIndex	Zero based input stream index
pBuffer	Buffer containing data Cannot be NULL
dwFlags	Must be a combination of the following flag valus (or 0) DMO_INPUT_DATA_BUFFERF_TIME **rtTimestamp** is valid DMO_INPUT_DATA_BUFFERF_TIMELENGTH **rtTimeLength** is valid DMO_INPUT_DATA_BUFFERF_SYNCPOINT Syncpoint at the beginning of the data
rtTimeStamp	Start timestamp in 100ns units
rtTimeLength	Length in 100ns units

Return Values

S_OK	Processed normally
S_FALSE	No output.
DMO_E_NOTACCEPTING	Data cannot be accepted until previous output has been processed by calling **ProcessOutput()**

HRESULT ProcessOutput(DWORD dwReserved, DWORD cOutputBufferCount, DMO_OUTPUT_DATA_BUFFER *pOutputDataBuffers, DWORD *pdwStatus)

Generate outputs from current input data. The status fields in the output data buffers are updated as a result of this call.

The IMediaBuffer pointers in the DMO_OUTPUT_DATA_BUFFER structures should not be held by AddRef after return from this call (ie their reference counts should be the same on exit as on entry).

Output buffer status fields are undefined if this call returns a failure code.

After calling **ProcessOutput()** the application should check all output streams for the DMO_OUTPUT_DATA_BUFFERF_INCOMPLETE flag. It is possible, for instance, when there are multiple output streams, for a stream which did not report DMO_OUTPUT_DATA_BUFFERF_INCOMPLETE previously to report it after a subsequent call to **ProcessOutput()**.

Parameters

dwFlags	DMO_PROCESS_OUTPUT_DISCARD_WHEN_NO_BUFFER If the pBuffer member of one of the output buffer structures is NULL discard output data.
cOutputBufferCount	Count of input buffers - this should be the same as the number of output streams.
pOutputDataBuffers	Array of output data buffers of size cOutputBufferCount.
pdwStatus	The Media Object should return 0 here.

Return Values

S_OK	Processing was successful
Failure code	Failure in processing

HRESULT GetInputMaxLatency(DWORD dwInputStreamIndex, REFERENCE_TIME *prtMaxLatency)

Returns the maximum latency in time between input on the stream and the corresponding output timestamps. Thus, for example, if input timestamped at time T generates output for time T-Delta then this value is the maximum possible value of Delta for the media types defined. This value does not take into account input buffer size.

Parameters

dwInputStreamIndex	0-based input stream index
prtMaxLatency	Latency

Return Values

E_NOTIMPL	Not implemented. Assume 0 latency
S_OK	OK
Failure code	Failure

HRESULT SetInputMaxLatency(DWORD dwInputStreamIndex, REFERENCE_TIME rtMaxLatency)

Sets the maximum latency in time between input on the stream and the corresponding output timestamps. Thus, for example, if input timestamped at time T generates output for time T-Delta then this bounds the maximum possible value of Delta for the media types defined. This value does not take into account input buffer size.

Parameters

dwInputStreamIndex	0-based input stream index
prtMaxLatency	Latency

Return Values

E_NOTIMPL	Not implemented. Latency cannot be set.
S_OK	OK
E_FAIL	Latency cannot be set

HRESULT Lock(LONG lLock)

Acquire a lock so that multiple operations can be performed while keeping the Media Object serialized.

Parameters

LLock	**TRUE** – lock **FALSE** - unlock
prtMaxLatency	Latency

Return Values

S_OK	OK
E_FAIL	Cannot lock

Registration

```
DMO_PARTIAL_MEDIATYPE
    {
        GUID type;
        GUID subtype;
    }
```

type	Major type for matching corresponding media types GUID_NULL means match any type
subtype	Subtype for matching corresponding media types GUID_NULL means match any subtype

HRESULT DMORegister(LPCWSTR szName, REFCLSID rclsidDMO, REFGUID rguidCategory, DWORD dwFlags, DWORD cInTypes, const DMO_PARTIAL_MEDIATYPE *pInTypes, DWORD cOutTypes, const DMO_PARTIAL_MEDIATYPE *pOutTypes)

Register a new object,its category and the media types it supports.

Parameters

szName	Registration name for this DMO. Names longer than 79 characters may be truncated.
rclsidDMO	Class ID the corresponding COM object for the DMO is registered under.
rclsidDMO	Class ID the corresponding COM object for the DMO is registered under.
rguidCategory	Category of this object
dwFlags	This must be a combination of the following flag values (or 0). DMO_REGISTERF_IS_KEYED Object use is restricted to by key
cInTypes	Number of input types to register

pInTypes	Input types
cOutTypes	Number of output types to register
pOutTypes	Output types

Return Values

HRESULT DMOUnregister(REFCLSID rclsidDMO, REFGUID rguidCategory)

Unregister a media object from one or all categories.

Parameters

rclsidDMO	Class ID of the DMO
rguidCategory	Remove from this category If this is GUID_NULL unregister this object from all categories

Return Values

HRESULT DMOEnum(REFGUID rguidCategory, DWORD dwFlags, DWORD cInTypes, const DMO_PARTIAL_MEDIA_TYPE *pInTypes, DWORD cOutTypes, const DMO_PARTIAL_MEDIA_TYPE *pOutTypes)

Enumerate Media Objects by category and/or by media type. GUID_NULL means match any GUID.

Parameters

rclsidDMO	Class ID the corresponding COM object for the DMO is registered under.
rguidCategory	Category of this object
dwFlags	This must be a combination of the following flag values (or 0). DMO_REGISTERF_INCLUDE_KEYED Include keyed objects
cInTypes	Number of input types to register
pInTypes	Input types
cOutTypes	Number of output types to register
pOutTypes	Output types

Return Values

Media Type helpers

Use these functions to manipulate media types.

Media types initialized with **MoInitMediaType** must be freed with **MoFreeMediaType**.
Media types created with **MoCreateMediaType** must be freed with **MoDeleteMediaType**.
Media types copied using **MoCopyMediaType** must be freed using **MoFreeMediaType**.
Media types duplicated using **MoDuplicateMediaType** must be freed using **MoDeleteMediaType**.

HRESULT MoInitMediaType(DMO_MEDIA_TYPE *pmt, DWORD cbFormat)

Initialize a media type with a given size format block. **pmt** is assumed uninitialized on input and no attempt is made to free any media type previously in **pmt**.

Parameters

pmt	Where to initialize the media type
cbFormat	Size of format block to create

Return Values

E_OUTOFMEMORY

HRESULT MoFreeMediaType(DMO_MEDIA_TYPE *pmt)

Free a media type previously initialized by **MoInitMediaType**.
On return the **pbFormat** field will be 0.

Parameters

pmt	Media type to free

Return Values

HRESULT MoCopyMediaType(DMO_MEDIA_TYPE *pmtDest, const DMO_MEDIA_TYPE *pmtSource)

Copy media types.

Parameters

pmtDest	Destination Media Type
pmtSource	Source Media Type

Return Values

E_OUTOFMEMORY	Could not allocate memory

HRESULT MoCreateMediaType(DMO_MEDIA_TYPE **ppmt, DWORD cbFormat)

Create a new media type structure.

Parameters

ppmt	Where to allocate the new media type
cbFormat	Size of format block

Return Values

E_OUTOFMEMORY

HRESULT MoDeleteMediaType(DMO_MEDIA_TYPE *pmt)

Delete a media type allocated by **MoCreateMediaType** or **MoDupicateMediaType()** or returned by pointer from an API or interface method.

Parameters

pmt	Media type to delete

Return Values

HRESULT MoDupliateMediaType(DMO_MEDIA_TYPE **ppmtDest, const DMO_MEDIA_TYPE *pmtSrc)

Duplicate a media type.

Parameters

ppmtDest	New type
pmtSrc	Source

Return Values

E_OUTOFMEMORY